Physics-Informed Neural Networks (PINNs)

UNIVERSITY OF OXFORD
Mathematical Institute

Dr Georg Maierhofer & Prof. Jared Tanner

Mathematical Institute
University of Oxford

Theories of Deep Learning: C6.5, guest lecture

Oxford Mathematics

Simulating physical systems with neural networks

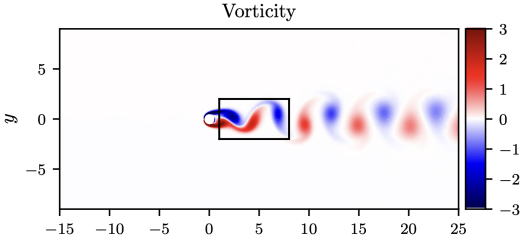


Figure: Vortex shedding behind a^{x}circular cylinder, Raissi et al. 2019

The dynamics is governed by the Navier–Stokes equations:

$$\mathbf{u} + \lambda_1(\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \lambda_2 \Delta \mathbf{u}.$$

Can the function approximation qualities of NN be used to simulate the dynamics?

https://www.sciencedirect.com/science/article/abs/pii/S0021999118307125

Two possible modalities of using neural networks for PDEs

Consider an abstract differential equation (for time-dependent problems $\Omega = [0, T] \times \tilde{\Omega}$):

$$\begin{cases} \mathcal{D}[u; \lambda] = f, & x \in \Omega, \\ u(x) = g(x), & x \in \partial\Omega. \end{cases} \tag{1}$$

Two types of problems for which NNs can be helpful:

- Given fixed model parameters λ what can be said about the unknown hidden state u of the system?
- What are the parameters λ that best describe the observed data?

By using Neural Networks as basis for the approximation, one might expect to achieve reduction in number of degrees of freedom based on exponential expressivity (cf. Lectures 3 & 4). Moreover, the nature in which training results in a "soft" constraint allows for easy combination of data and physics.

So far focussed on architecture, what about design of loss function? With given training data can enforce observations with MSE:

$$\mathcal{L}_{data}(\theta) = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} |u(x_i) - u_i|^2, \qquad m\frac{d^2u}{dx^2} + \mu\frac{du}{dx} + ku = 0.$$

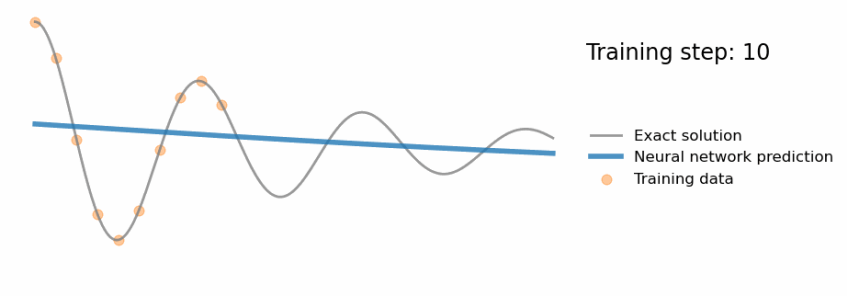


Figure: The generalisation error of supervised learning. Source: Dr Ben Moseley's blog

Data-driven inference - supervised learning

So far focussed on architecture, what about design of loss function? With given training data can enforce observations with MSE:

$$\mathcal{L}_{data}(\theta) = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} |u(x_i) - u_i|^2, \qquad m\frac{d^2u}{dx^2} + \mu\frac{du}{dx} + ku = 0.$$

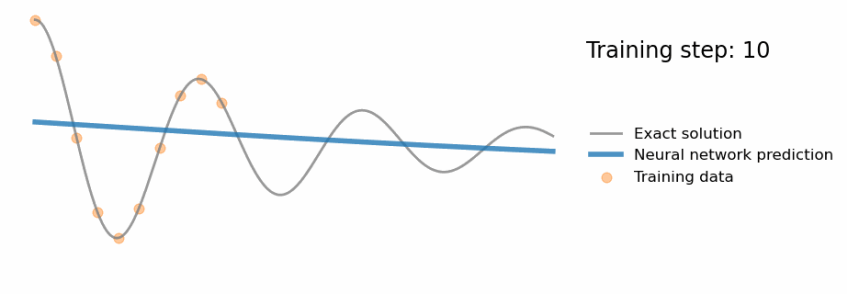


Figure: The generalisation error of supervised learning. Source: Dr Ben Moseley's blog

Data-driven inference - supervised learning

So far focussed on architecture, what about design of loss function? With given training data can enforce observations with MSE:

$$\mathcal{L}_{data}(\boldsymbol{\theta}) = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} |u(x_i) - u_i|^2, \qquad m\frac{d^2u}{dx^2} + \mu\frac{du}{dx} + ku = 0.$$

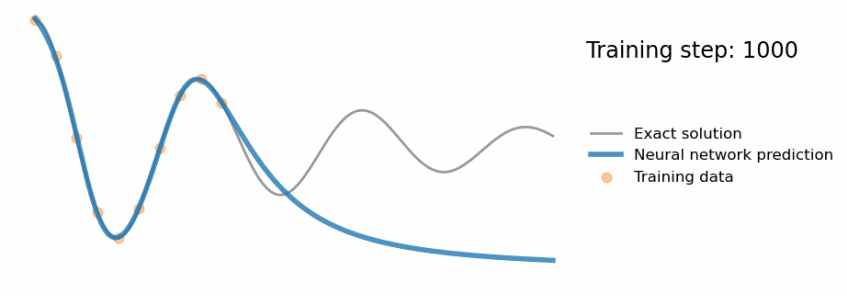


Figure: The generalisation error of supervised learning. Source: Dr Ben Moseley's blog

Incomplete data can be supplemented with physical knowledge.

Physics-informed loss, $\mathcal{L}_f$ enforces the structure imposed by equation (1):

$$\mathcal{L}_f(\boldsymbol{\theta}) = \frac{1}{N_{col}} \sum_{i=1}^{N_{col}} |\mathcal{D}[u](x_f^i) - f(x_f^i)|^2,$$

where $\{(x_f^i)\}_{i=1}^{N_{col}}$ represents the collocation points on Ω.

Note: Require NN to be sufficiently differentiable for this PDE residual to be well-defined, i.e. ReLu activation may not be allowed.

Typically have to impose also (initial) and boundary conditions:

$$\mathcal{L}_b := \frac{1}{N_b} \sum_{i=1}^{N_b} |u(x_b^i) - g(x_b^i)|^2,$$

with $\{(x_b^i)\}_{i=1}^{N_b} \subset \partial\Omega$.

Incomplete data can be supplemented with physical knowledge

Differential equation (damped oscillator): $m\frac{d^2u}{dx^2} + \mu\frac{du}{dx} + ku = 0.$

Mixed Loss: $\mathcal{L}(\boldsymbol{\theta}) = \alpha\mathcal{L}_{data}(\boldsymbol{\theta}) + (1-\alpha)\mathcal{L}_f(\boldsymbol{\theta}),\ \alpha \in (0,1).$

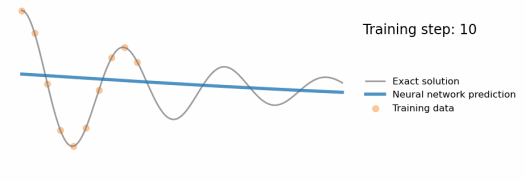


Figure: Data alone.

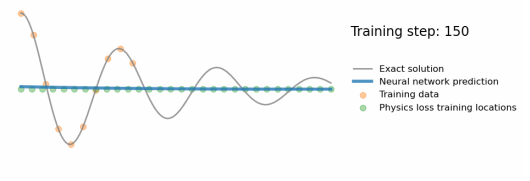


Figure: Data + Physics.

Source: Dr Ben Moseley's blog

Incomplete data can be supplemented with physical knowledge

Differential equation (damped oscillator): $m\frac{d^2u}{dx^2} + \mu\frac{du}{dx} + ku = 0.$

Mixed Loss: $\mathcal{L}(\boldsymbol{\theta}) = \alpha\mathcal{L}_{data}(\boldsymbol{\theta}) + (1-\alpha)\mathcal{L}_f(\boldsymbol{\theta}),\ \alpha \in (0,1).$

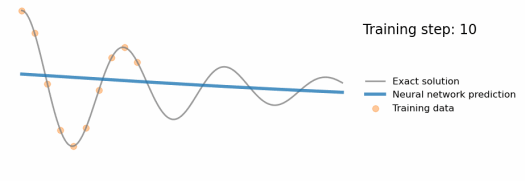


Figure: Data alone.

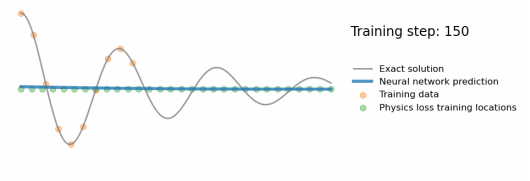


Figure: Data + Physics.

Source: Dr Ben Moseley's blog

Incomplete data can be supplemented with physical knowledge

Differential equation (damped oscillator): $m\frac{d^2u}{dx^2} + \mu\frac{du}{dx} + ku = 0.$

Mixed Loss: $\mathcal{L}(\boldsymbol{\theta}) = \alpha\mathcal{L}_{data}(\boldsymbol{\theta}) + (1-\alpha)\mathcal{L}_f(\boldsymbol{\theta}),\ \alpha \in (0,1).$

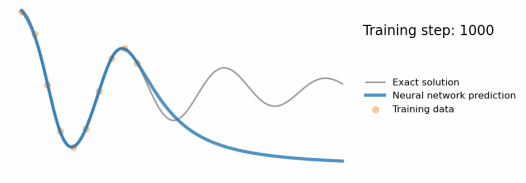


Figure: Data alone.

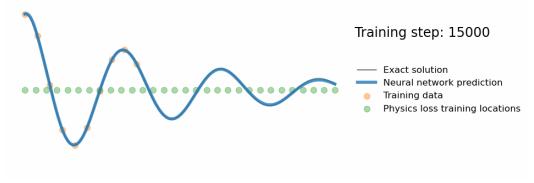


Figure: Data + Physics.

Source: Dr Ben Moseley's blog

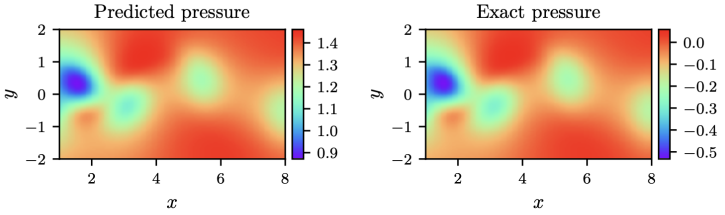
Example: Navier–Stokes equations:

$$u_t + \lambda_1 (u u_x + v u_y) = -p_x + \lambda_2 (u_{xx} + u_{yy})$$
$$v_t + \lambda_1 (u v_x + v v_y) = -p_y + \lambda_2 (v_{xx} + v_{yy})$$

Here, $\lambda = (\lambda_1, \lambda_2)$ are the unknown parameters (inverse of fluid density and the kinematic viscosity). Given noisy measurements $\left\{t^i, x^i, y^i, u^i, v^i\right\}_{i=1}^N$ of the velocity field, we are interested **in learning the parameters** λ as well as **the pressure** $p(t, x, y)$. For this choose NN which approximates velocity field ψ ($\partial_x \psi = u, \partial_y \psi = v$) and pressure field p and minimize the *mixed* loss:

$$\begin{aligned}\mathcal{L} := & \frac{1}{N} \sum_{i=1}^N \left(\left| u\left(t^i, x^i, y^i\right) - u^i \right|^2 + \left| v\left(t^i, x^i, y^i\right) - v^i \right|^2 \right) \\ & + \frac{1}{N} \sum_{i=1}^N \left| u_t + \lambda_1 (u u_x + v u_y) + p_x - \lambda_2 (u_{xx} + u_{yy}) \right|^2_{(t^i, x^i)} \\ & + \frac{1}{N} \sum_{i=1}^N \left| v_t + \lambda_1 (u v_x + v v_y) + p_y - \lambda_2 (v_{xx} + v_{yy}) \right|^2_{(t^i, x^i)} .\end{aligned}$$

Data-driven discovery of partial differential equations



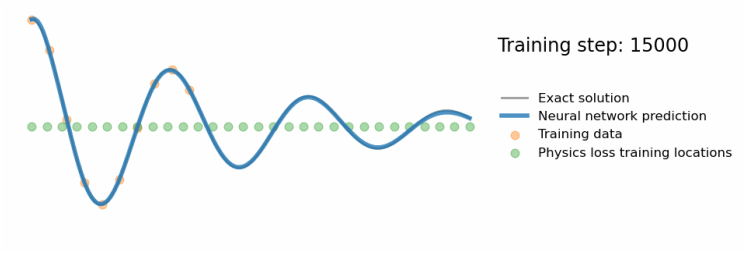
Correct PDE	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy})$ $v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
Identified PDE (clean data)	$u_t + 0.999(uu_x + vu_y) = -p_x + 0.01047(u_{xx} + u_{yy})$ $v_t + 0.999(uv_x + vv_y) = -p_y + 0.01047(v_{xx} + v_{yy})$
Identified PDE (1% noise)	$u_t + 0.998(uu_x + vu_y) = -p_x + 0.01057(u_{xx} + u_{yy})$ $v_t + 0.998(uv_x + vv_y) = -p_y + 0.01057(v_{xx} + v_{yy})$

Figure: Predicted vs exact pressure and PDE parameters (Source: Raissi et al. 2019)

Incomplete data can be supplemented with physical knowledge

Differential equation (damped oscillator): $m\frac{d^2u}{dx^2} + \mu\frac{du}{dx} + ku = 0.$

Mixed Loss: $\mathcal{L}(\boldsymbol{\theta}) = \alpha\mathcal{L}_{data}(\boldsymbol{\theta}) + (1-\alpha)\mathcal{L}_f(\boldsymbol{\theta}),\ \alpha \in (0,1).$



Generalisation error: $\int_0^T |u(x) - u_\theta^*(x)|dx$

The generalisation error of physics-informed neural networks

The error of PINNs on unseen points in the spatio-temporal domain can be controlled in terms of training error, if:

- The **solution of the underlying PDE is well-behaved** (with respect to perturbations of inputs) in a very precise manner.
- The **number of collocation** points is sufficiently large.
- Implicit constants that arise in the stability and quadrature error estimates, which depend on **the underlying PINNs, are controlled in a suitable manner**.

For the time being the precise architecture of the neural network will be secondary, and we focus on the physics-informed loss.

Slightly simplified from:
https://www.sciencedirect.com/science/article/abs/pii/S0021999118307125

Example - Poisson's equation with Dirichlet boundary conditions

Let $\Omega = [0,1] \subset \mathbb{R}$. For $f \in L^2([0,1];\mathbb{R})$ consider

$$\begin{cases} \partial_{xx} u = f, & x \in \Omega, \\ u = 0, & x \in \partial\Omega. \end{cases} \tag{2}$$

This is an example of a well-posed PDE with the following properties:

- **Existence and uniqueness of solutions:** for every $f \in L^2([0,1];\mathbb{R})$ there is a *unique* solution $u \in H_0^2(\omega)$ such that (2) is satisfied.
- **Stability:** By a standard (Poincaré) estimate:

$$\|\mathbf{u} - \mathbf{v}\|_{W^{2,2}(\Omega;\mathbb{R})} \leq C_\Omega \|\partial_{xx}\mathbf{u} - \partial_{xx}\mathbf{v}\|_{L^2(\Omega;\mathbb{R})}.$$

An abstract framework for PDE problems

Abstract formulation of PDE

$$\mathcal{D}(\mathbf{u}) = \mathbf{f}, \tag{3}$$

where $\mathcal{D} : X^* \mapsto Y^*$ with $X^* \subset W^{s,q}(\Omega; \mathbb{R}^m)$ is a bounded operator and, $Y^* \subset L^p(\Omega; \mathbb{R}^m)$ are closed subspaces, for some $m \geqslant 1, 1 \leq p, q < \infty$ and $s \geqslant 0$. Note $\Omega = (0, T) \times \Omega_{space}$ for time-dependent problems.

In previous example: $X = H^2([0,1]), X^* = H_0^2([0,1]), Y^* = Y = L^2([0,1])$.

Well-posedness

- Existence and uniqueness of solutions: $\mathcal{D} : X^* \to Y^*$ is a bijection.
- Stability: For any $\mathbf{u}, \mathbf{v} \in X^*$, the differential operator $\mathcal{D}$ satisfies

$$\|\mathbf{u} - \mathbf{v}\|_X \leq C_{\text{pde}} \|\mathcal{D}(\mathbf{u}) - \mathcal{D}(\mathbf{v})\|_Y \tag{4}$$

Here, the constant $C_{\text{pde}} > 0$ can explicitly depend on $\|\mathbf{u}\|_Z$ and $\|\mathbf{v}\|_Z$, for some norm $\|\cdot\|_{Z_X}$ on a subspace $Z_X \subset X^*$.

PINNs loss and generalisation error

- *PDE Residual for neural network approximation:* $\mathcal{R}_\theta = \mathcal{R}(\mathbf{u}_\theta^*) := \mathcal{D}(\mathbf{u}_\theta^*) - \mathbf{f} \in Y^*$.
- *Training/collocation points for PINN-loss:* $\mathcal{S} = \{y_n\}_{n=1}^N \subset \Omega$.
- We monitor the *training error* given by (recall $Y^* \subset L^2([0,1])$):

$$\mathcal{E}_T := \left(\sum_{n=1}^{N} w_n \left| \mathcal{R}_{\theta^*}(y_n) \right|^p \right)^{\frac{1}{p}}$$

Definition (Generalisation error)

The error of approximating the solution $\mathbf{u}$ of (3) by the PINN $\mathbf{u}_\theta^*$,

$$\mathcal{E}_G = \mathcal{E}_G(\theta^*; \mathcal{S}) := \|\mathbf{u} - \mathbf{u}_\theta^*\|_X = \|\mathbf{u} - \mathbf{u}_\theta^*\|_{W^{s,q}}.$$

Goal: Relate the generalisation error (unknown!) to the training error (known!).

Choice of collocation points - numerical quadrature

To this end, we consider a generic mapping $g : \Omega \mapsto \mathbb{R}^m$ with $g \in Y^*$. We are interested in approximating the integral,

$$\bar{g} := \int_\Omega g(y)\mathrm{d}y,$$

with $\mathrm{d}y$ denoting the Lebesgue measure. In order to approximate the above integral by a quadrature rule, we need the quadrature points $y_i \in \Omega$ for $1 \leq i \leq N$ for some $N \in \mathbb{N}$ as well as weights w_i with $w_i \in \mathbb{R}_+$. Then a quadrature is defined by

$$\bar{g}_N := \sum_{i=1}^{N} w_i g\left(y_i\right)$$

for weights w_i and quadrature points y_i.

Assumption (Quadrature error)

We take a quadrature rule for which the error is bounded as

$$|\bar{g} - \bar{g}_N| \leq C_{\text{quad}} \ \left(\|g\|_{Z_Y}, \bar{d}\right) N^{-\alpha},$$

for some $\alpha > 0$, where $\|\cdot\|_{Z_Y}$ is a norm on a suitable subspace $Z_Y \subset Y$.

Example: Trapezoidal rule on $\Omega = [0, 1]$. Take $x_i = (i-1)/(N-1), i = 1, \ldots, N$ and $w_1 = w_N = 1/(2(N-1)), w_i = 1/(N-1), i = 2, \ldots, N-1$. Then the quadrature error satisfies the following bound:

$$|\bar{g} - \bar{g}_N| \leq \frac{1}{12} \|g''\|_{L^\infty} N^{-2}.$$

Theorem

Let $\mathbf{u} \in Z_X \subset X^*$ be the unique solution of the PDE with all above assumptions. Let $\mathbf{u}^* \in Z_X \subset X^*$ be the PINN. Further assume that the residual $\mathcal{R}_{\theta^*} \in Z_Y$ and that the quadrature assumption is satisfied. Then the following estimate on the generalization error holds:

$$\mathcal{E}_G \leq C_{\mathrm{pde}} \mathcal{E}_T + C_{\mathrm{pde}} C_{\mathrm{quad}}^{\frac{1}{p}} N^{-\frac{\alpha}{p}},$$

with constants $C_{pde} = C_{pde}\left(\|\mathbf{u}\|_{Z_X}, \|\mathbf{u}^*\|_{Z_X}\right)$ and $C_{quad} = C_{quad}\left(\||\mathcal{R}_{\theta^*}|^p\|_{Z_Y}\right)$. Note that these constants $C_{\mathrm{pde}}, C_{quad}$ depend on the underlying PINN $\mathbf{u}^*$, which in turn can depend on the number of training points N.

Estimating the generalisation error

The theorem tells us that the generalisation error for the PINN is small as long as:

- **The training error $\mathcal{E}_T$ is sufficiently small.** Note that we have no a priori control on the training error but can compute it *a posteriori*.
- **The quadrature error of the PDE residual is small**, i.e. we have sufficiently many well-chosen collocation points.
- **The PDE is well-posed:** this is encoded in the constant C_{pde}.
- **The Neural Network is well-behaved:** this is encoded in C_{pde} and C_{quad}.

The above constants depend on the details on the underlying PDE and quadrature rule and cannot be worked out in the abstract setup here but can be computed in specific cases.

Proof. We can directly apply the stability bound (4) to the generalisation error to find

$$\begin{aligned}\mathcal{E}_G &= \|\mathbf{u} - \mathbf{u}^*\|_X \\ &\leq C_{\text{pde}} \, \|\mathcal{D}(\mathbf{u}^*) - \mathcal{D}(\mathbf{u})\|_Y \\ &\leq C_{\text{pde}} \, \|\mathcal{R}\|_Y\end{aligned}$$

where we recall $\mathcal{R} = \mathcal{R}_{\theta^*} = \mathcal{D}(\mathbf{u}^*_\theta) - \mathbf{f}$, is the residual corresponding to the trained neural network $\mathbf{u}^*$ and we used that

$$\mathcal{D}(\mathbf{u}) = f.$$

By the fact that $Y = L^p(\Omega)$, the definition of the training error and the quadrature rule, we see that

$$\|\mathcal{R}\|_Y^p \approx \left(\sum_{n=1}^{N} w_n \, |\mathcal{R}_{\theta^*}|^p \right) = \mathcal{E}_T^p.$$

Estimating the generalisation error

Hence, the training error is a quadrature for the residual and the resulting quadrature error translates to

$$\|\mathcal{R}\|_Y^p \leq \mathcal{E}_T^p + C_{\text{quad}}\, N^{-\alpha}.$$

Therefore,

$$\begin{aligned}\mathcal{E}_G^p &\leq C_{\text{pde}}^p \|\mathcal{R}\|_Y^p \\ &\leq C_{\text{pde}}^p \left(\mathcal{E}_T^p + C_{\text{quad}} N^{-\alpha}\right).\end{aligned}$$

Thus,

$$\mathcal{E}_G \leq C_{\text{pde}} \mathcal{E}_T + C_{\text{pde}} C_{\text{quad}}^{\frac{1}{p}} N^{-\frac{\alpha}{p}}$$

which is the desired estimate.

□

Choice of collocation points in PINNs

The above theorem shows that the generalisation error is strongly dependent on the quadrature error of the PDE residual $\mathcal{R}_{\theta^*}$ in $\|\cdot\|_Y$, i.e.

$$\|\mathcal{R}\|_Y^p \approx \left(\sum_{n=1}^{N} w_n \left|\mathcal{R}_{\theta^*}\right|^p\right) = \mathcal{E}_T^p.$$

Equidistribution principle: Consider simple example, $\int_a^b f(x)\mathrm{d}x$. Using trapezoidal rule we have for $a = x_1 < \cdots < x_N = b$

$$\int_a^b f(x)\mathrm{d}x \approx \sum_{i=1}^{N-1} \frac{x_{i+1} - x_i}{2}(f(x_i) + f(x_{i+1})), \quad \text{Error} = \sum_{i=1}^{N-1} \frac{(x_{i+1} - x_i)^2}{12} \max_{x\in[x_i, x_{i+1}]} |f''(x)|.$$

This suggests we want $x_{i+1} - x_i$ small when $\max_{x\in[x_i,x_{i+1}]} |f''(x)|$ is large, i.e. we want many collocation points in regions where the PDE residual varies rapidly. $\rightarrow$ if we have access to residual can sample collocation points adaptively!

Adaptive collocation point sampling in PINNs

Residual-based Adaptive Distribution (RAD, Wu et al. 2023): Sample collocation points according to the distribution

$$p(x) \propto \frac{|\mathcal{R}_\theta(x)|^k}{\mathbb{E}\left[|\mathcal{R}_\theta(x)|^k\right]} + c \tag{5}$$

where $\mathcal{R}_\theta(x) = \mathcal{D}(u_\theta^*)(x) - f(x)$. Then repeatedly update collocation points during training:

0. Choose initial set of collocation points (e.g. uniform sampling);
1. Train the PINN for a certain number of iterations;
2. Sample new set of collocation points according to (5);
3. Repeat 1. & 2.

Residual-based Adaptive Distribution (RAD)

Convection equation: $\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0, \quad x \in [0,1], t \in [0,T], \quad u(x,0) = h(x), \ x \in [0,1].$

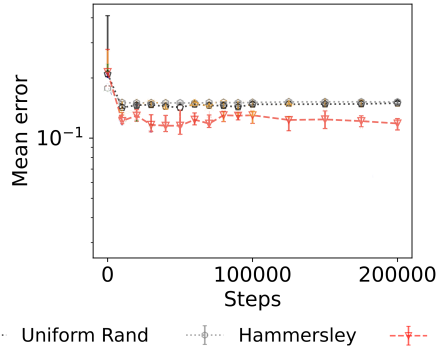


Figure: Generalisation error during PINN training (Source: Lau et al. 2024).

Failure modes of PINNs

So far seen improvements on generalisability through physics-informed loss function. In light of lectures 9 & 10 should also look at the effect on the optimisation landscape.

Test case: 1d convection

$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0, \quad x \in \Omega, t \in [0, T], \quad u(x, 0) = h(x), \quad x \in \Omega.$$

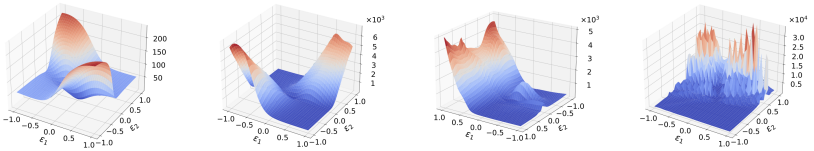
Here, β is the *convection coefficient*. The general loss function for this problem is

$$\mathcal{L}(\theta) = \underbrace{\frac{1}{N_u} \sum_{i=1}^{N_u} \left(\hat{u} - u_0^i\right)^2}_{\text{inital data}} + \underbrace{\frac{1}{N_f} \sum_{i=1}^{N_f} \lambda_i \left(\frac{\partial \hat{u}}{\partial t} + \beta \frac{\partial \hat{u}}{\partial x}\right)^2}_{\text{PDE}} + \underbrace{\frac{1}{N_b} \sum_{i=1}^{N_b} (\hat{u}(\theta, 0, t) - \hat{u}(\theta, 2\pi, t))^2}_{\text{boundary data}}$$

where $\hat{u} = NN(\theta, x, t)$ is the output of the NN.

https://proceedings.neurips.cc/paper/2021/file/df438e5206f31600e6ae4af72f2725f1-Paper.pdf

PINNs loss landscape



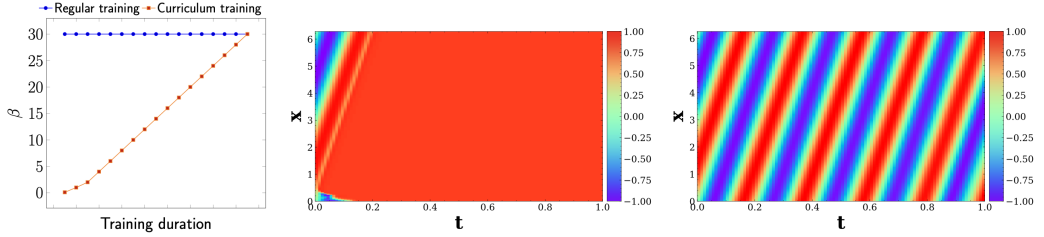
(a) $\beta = 1.0$ (b) $\beta = 10.0$ (c) $\beta = 20.0$ (d) $\beta = 30.0$

β	1	10	20	30
Relative error	7.84×10^{-3}	1.08×10^{-2}	7.50×10^{-1}	8.97×10^{-1}
Absolute error	3.17×10^{-3}	6.03×10^{-3}	4.32×10^{-1}	5.42×10^{-1}

Figure: Loss landscapes for varying values of convection coefficient.

Observation: Non-trivial convection regime leads to more challenging optimisation landscapes and reduces PINN approximation quality. PDE solutions "less regular"...

Transfer learning in PINNs (cf lecture 11)



(a) *Curriculum regularization schematic*

(b) *Regular training PINN solution for* $\beta = 30$

(c) *Curriculum training PINN solution for* $\beta = 30$

Figure: Performance of curriculum training on PINN (source: Krishnapriyan et al 2021)

Observation: Instead of training the PINN to learn the solution right away for cases with higher β, we start by training the PINN on lower β (easier for the PINN to learn), then gradually move to training the PINN on higher β.

Adaptive collocation point sampling revisited (see Lau et al. 2024)

Recall from lecture 9 (and neural tangent kernel theory) that for small stepsizes η in gradient descent the training dynamics at training step t of the PINN is approximately linear

$$\mathcal{R}[u^*_{\theta_{t+1}}](x) - \mathcal{R}[u^*_{\theta_t}](x) \approx -\eta \Theta_t(x, x')\mathcal{R}[u^*_{\theta_t}](x'),$$

where

$$\Theta_t(x, x') = \nabla_\theta \mathcal{R}[u^*_{\theta_t}](x) \nabla_\theta \mathcal{R}[u^*_{\theta_t}](x')^\top.$$

To optimise training dynamics, we would want to maximise the change in residual on the collocation points, i.e. the quantity

$$\Delta\mathcal{R}_{\theta_t}(x, \mathcal{S}) = \mathcal{R}_{\theta_{t+1}}(x; \mathcal{S}) - \mathcal{R}_{\theta_t}(x).$$

Instead of considering this for individual points, would be preferrable to have a measure for change in $\mathcal{R}$ for a given set $\mathcal{S}$ of collocation points.

Adaptive collocation point sampling revisited (see Lau et al. 2024)

For this we fix a set of collocation points $\mathcal{S}_t$. Then we perform one step of gradient descent, and estimate the empirical neural tangent kernel

$$\Theta_t(x, x').$$

Finally, we can consider the inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}_{\Theta_t}}$ associated with this kernel and define

$$\alpha(\mathcal{S}_t) := \langle \Delta\mathcal{R}_{\theta_t}(\cdot, \mathcal{S}_t), \Delta\mathcal{R}_{\theta_t}(\cdot, \mathcal{S}_t) \rangle_{\mathcal{H}_{\Theta_t}},$$

where, for $f_1, f_2 : \Omega \to \mathbb{R}$, $\langle f_1(\cdot), f_2(\cdot) \rangle_{\mathcal{H}_{\Theta_t}} = \int_{\Omega\times\Omega} \frac{f_1(x) f_2(x')}{\Theta_t(x,x')} dx dx'$.

Refined generalisation error estimate (see Lau et al. 2024)

Theorem

Suppose $\Omega = [0,1]^d$ and $\mathcal{S} \subset \Omega$ is an i.i.d. sample of size N_S from a fixed distribution. Let $\hat{u}_\theta$ be a NN which is trained on $\mathcal{S}$ by GD to convergence, with a small enough learning rate such that Θ_t remains constant. Then, there exists constants $c_1, c_2, c_3 = \mathcal{O}\left(\text{poly}\left(1/N_S, \lambda_{\max}\left(\Theta_{0,\mathcal{S}}\right)/\lambda_{\min}\left(\Theta_{0,\mathcal{S}}\right)\right)\right)$ such that with high probability over the random model initialization and the sample $\mathcal{S}$,

$$\int_\Omega |\hat{u}_{\theta_\infty}(x) - u(x)|\, dx \leq c_1 \left\|\mathcal{R}_{\theta_\infty}(\cdot)\right\|_{\ell^1(\mathcal{S})} + c_2 \alpha(\mathcal{S})^{-1/2} + c_3.$$

$\rightarrow$ If training and architecture is taken into account (captured by Θ_t) then generalisation error is smallest when $\alpha(\mathcal{S})$ is largest.

PINNACLE (PINN Adaptive ColLocation and Experimental points selection)

0. Initialise the NN with θ_0.
1. Estimate the eigenfunctions $\psi_{t,i}$ and eigenvalues $\lambda_{t,i}$ of Θ_t:

$$\Theta_{t,\mathcal{S}_t}(x, x') = \sum_{i=1}^{\infty} \lambda_{t,i} \psi_{t,i}(x) \psi_{t,i}(x').$$

2. Observe that for any $\tilde{S}$ (even of size one):

$$\alpha(\tilde{S}) = \langle \Delta\mathcal{R}_{\theta_t}(\cdot, \tilde{S}), \Delta\mathcal{R}_{\theta_t}(\cdot, \tilde{S}) \rangle_{\mathcal{H}_{\Theta_t}} = \sum_{i=1}^{\infty} \lambda_{t,i} \underbrace{|\langle \psi_{t,i}(\tilde{S}), \mathcal{R}_{\theta_t}(\tilde{S}) \rangle|^2}_{\text{inner product of vectors}}.$$

3. Sample collocation points S_{t+1} from the estimated density

$$p(x) \propto \hat{\alpha}(x), \text{ where } \hat{\alpha}(x) = \alpha(\{x\}).$$

4. Take a gradient descent step and repeat 1.-3.

PINNACLE (PINN Adaptive ColLocation and Experimental points selection)

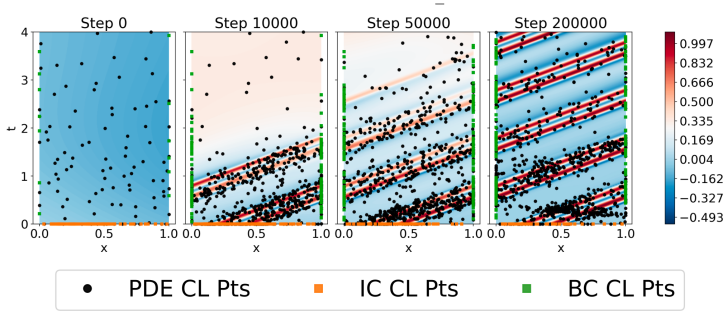


Figure: Adaptive collocation point selection for PINN (Source: Lau et al. 2024)

Convection equation: $\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0, \quad x \in [0,1], t \in [0,T], \quad u(x,0) = h(x), \ x \in [0,1].$

PINNACLE (PINN Adaptive ColLocation and Experimental points selection)

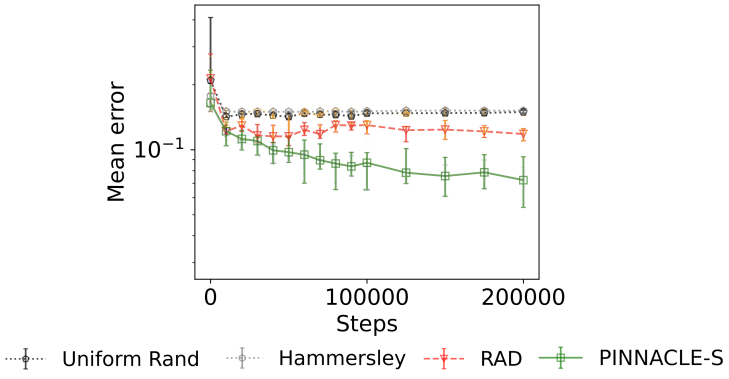


Figure: Generalisation error during PINN training (Source: Lau et al. 2024)

Further topics in deep learning for scientific computing

- **Operator learning:**
  Boullé, N. & Townsend, A. (2024) 'A mathematical guide to operator learning', Handbook of Numerical Analysis.

  Lu, L. et al. (2021) 'Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators', Nature Machine Intelligence.
- **Large-scale Spatiotemporal Forecasting:**
  Bi, K. et al. (2023) 'Pangu-Weather: Accurate medium-range global weather forecasting with 3D neural networks', Nature.

  Lam, R. et al. (2023) 'GraphCast: Learning Skillful Medium-Range Global Weather Forecasting', Science.
- **Hybrid methods:**
  Song, W. et al. (2022) 'M2N: Mesh Movement Networks for PDE Solvers', NeurIPS.